Paper submitted at the International journal of computational fluid dynamics, 2016

NEW PARALLEL METHOD FOR ADJACENT DISCONNECTED UNSTRUCTURED 3D MESHES

J. MUELA*, D. MARTÍNEZ*, O. LEHMKUHL*,†, C.D. PÉREZ-SEGARRA* AND A. OLIVA*

* Heat and Mass Transfer Technological Center (CTTC)
Universitat Politècnica de Catalunya
ETSEIAT, Colom 11, 08222, Terrassa, Barcelona, Spain
E-mail: cttc@cttc.upc.edu, Web page: http://www.cttc.upc.edu

†Termo Fluids S.L.
Avinguda Jaquard, 97 1-E, 08222, Terrassa, Barcelona, Spain
E-mail: termofluids@termofluids.com, Web page: http://www.termofluids.com

Key words: Dynamic Mesh, ALE, Large Eddy Simulation, Wind Turbines

Abstract. A new parallel method for simulations with non-overlapping disconnected mesh domains but adjacent boundaries is presented and studied. This technique allows simulations using 3D unstructured meshes that are independent.

1 INTRODUCTION

In Computational Fluid Dynamics (CFD), traditionally, an Eulerian framework is employed since the majority of the problems of interest are studied in a static volume of the space. But this traditional approximation may not be the best option for some specific cases, like for example in Fluid-Structure Interaction (FSI) problems. Consider a wind turbine: it would be useful to work with a mesh *attached* to the rotating blades (Eulerian-Lagrangian framework), but also a static mesh would be desirable for the tower and the nacelle (Eulerian framework). The same logic could be applied to other similar problems like turbo-machinery. Therefore, it is of clear interest to develop a method able to perform simulations where a mesh domain Ω_a is moving with respect to a second static mesh domain Ω_b. Both domains are not overlapped $\Omega_a \cap \Omega_b = 0$ but they are adjacent and share some or all boundaries $\partial\Omega_{a_i} = \partial\Omega_{b_i}$

2 METHODOLOGY

The aim of the developed methodology is to *stitch* two independent meshes Ω_a and Ω_b that share a common boundary $\partial\Omega_{a_i} = \partial\Omega_{b_i}$. In order to do that, the implemented technique works in the following way: at the beginning of the simulation, each boundary cell of $\partial\Omega_{a_i}$ places a *mirror* node in mesh Ω_b, and vice versa. This mirror node is a lagrangian particle that is moved with the relative velocity between both meshes in order to preserve the relative position between the parent cell and the mirror node (see Fig. 1). This mirror node allows to easily interpolate the value of any scalar or gradient on the

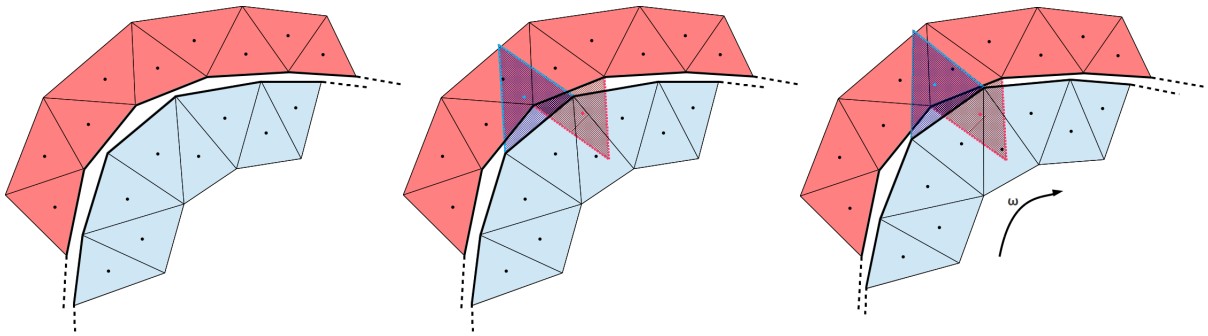


Figure 1: Schematic representation of a mirror node in the sliding boundary.

neighbouring domain and transfer the information to the boundary node. Furthermore, this mirror node is employed to reconstruct at each iteration the topology of the Poisson matrix.

2.1 Mathematical formulation

The Navier-Stokes equations for incompressible flows in an Arbitrary Lagrangian Eulerian (ALE) formulation are:

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \left(\left(\mathbf{u} - \mathbf{v}_d \right) \mathbf{u} \right) = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u} \tag{2}$$

where $\mathbf{u}$ is the velocity vector, ρ the density, p the pressure, ν the kinematic viscosity and $\mathbf{v}_d$ the displacement velocity of the domain with respect to the Eulerian reference framework. The Finite-Volume Method (FVM) is employed to discretize equations (1) and (2) on a general arbitrary mesh. Moreover, the velocity-pressure coupling is solved by means of the Fractional Step Method (FSM):

$$\mathbf{u}^p = \mathbf{u}^n + \Delta t \left[-\nabla \cdot \left(\mathbf{u}^n \mathbf{u}^n \right) - \nabla \cdot \left(\mathbf{v}_d^n \mathbf{u}^n \right) + \nu \nabla^2 \mathbf{u}^n \right] \tag{3}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^p - \frac{\Delta t}{\rho} \nabla p^{n+1} \tag{4}$$

where the superscript n refers to the time step, $\mathbf{u}^p$ is the predicted velocity and Δt the time step. The convective and the diffusive terms are solved explicitly, while the pressure is solved implicitly in order to guarantee that $\mathbf{u}^{n+1}$ is divergence-free. Using a collocated mesh scheme, the predictor step equation is discretized integrating Eq. (3) over a control volume c and applying the divergence theorem to its faces.

$$\mathbf{u}_c^p = \mathbf{u}_c^n + \frac{\Delta t}{V_c} \left[-\sum_{f \in F(c)} \mathbf{u}_f^n \hat{U}_f^n A_f - \sum_{f \in F(c)} \mathbf{u}_f^n \hat{V}_{df}^n A_f + \nu \sum_{f \in F(c)} \left(\mathbf{u}_{nb}^n - \mathbf{u}_c^n \right) \frac{A_f}{\delta d_f} \right] \tag{5}$$

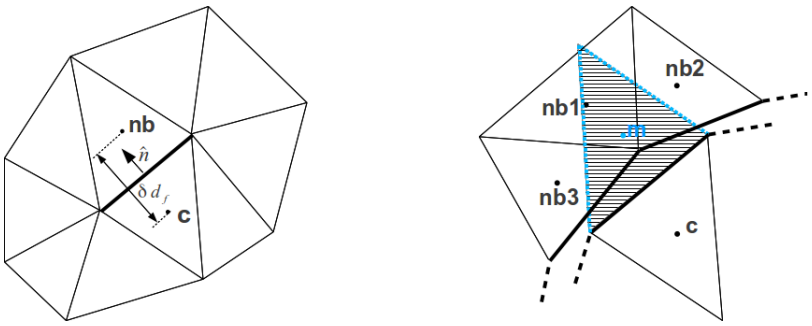


Figure 2: 2D Full (left) and Sliding (right) mesh detail.

where V_c is the volume of cell c, $\mathbf{u}_f$ is the velocity vector at the face f, $\hat{U}_f$ the normal face velocity, A_f the surface of f, the subscript nb refers to the face-neighbouring cells of c, δd_f is the distance between the cell centroids of c and nb projected onto the normal of the face f, and $\hat{V}_{df}$ is the normal face displacement velocity respect the reference frame. Applying the divergence operator $\nabla\cdot$ to Eq. (4) and imposing the divergence-free condition of Eq. (1) the discretized Poisson equation employed to solve the pressure reads:

$$\sum_{f\in F(c)} \hat{U}_f^p A_f = \frac{\Delta t}{\rho} \sum_{f\in F(c)} \left(p_{nb}^{n+1} - p_c^{n+1}\right) \frac{A_f}{\delta d_f} \tag{6}$$

Once the solution of p^{n+1} is obtained, the discrete form of Eq. (4) is used to calculate $\mathbf{u}^{n+1}$:

$$\mathbf{u}_c^{n+1} = \mathbf{u}_c^p - \frac{\Delta t}{\rho V_c} \sum_{f\in F(c)} p_f^{n+1} \hat{\mathbf{n}}_f A_f \tag{7}$$

where $\hat{n}_f$ is the unit normal vector of face f pointing outwards and p_f the pressure interpolated at the face f. The values for velocity and pressure at the face $(\mathbf{u}_f, \hat{U}_f, p_f)$ have to be interpolated. The face velocity is calculated using a symmetry-preserving scheme $\mathbf{u}_f = \frac{1}{2}(\mathbf{u}_c + \mathbf{u}_{nb})$. The normal face velocity and the face pressure are calculated as $\hat{U}_f^n = \frac{1}{2}(\mathbf{u}_c + \mathbf{u}_{nb}) \cdot \hat{\mathbf{n}}_f$ and $p_f = \frac{1}{2}(p_c + p_{nb})$ in order to minimize the kinetic energy conservation error [2]. The normal face displacement velocity is $\hat{V}_{df} = \mathbf{v}_{df} \cdot \hat{\mathbf{n}}_f$ (i.e. in rotating meshes $\mathbf{v}_{df} = \boldsymbol{\omega}_f \times (\mathbf{x}_f - \mathbf{x}_0)$, where $\boldsymbol{\omega}_f$ is the rotating velocity, $\mathbf{x}_f$ the face centroid and $\mathbf{x}_0$ the centre of rotation). It is obvious that the presented methodology cannot be applied straightforwardly in the sliding face, since as can be seen in Fig. 2, in the sliding face two *neighbouring* cells are not connected by a unique shared face f. As explained before, each cell with a face in the common boundary $\partial\Omega_{a_i} = \partial\Omega_{b_i}$ places a mirror node in the neighbour mesh. Each mirror node is assumed to belong to a cell that is a mirror of the parent cell. This assumption allows this face (f_b) to be treated as a pseudo-inner face between the cells c and m (see Fig. 2), which at the same time allows the calculation of the face values for velocity and pressure in a similar fashion to the inner faces. For the explicit operators, i.e. the convective and the diffusive operators, the value

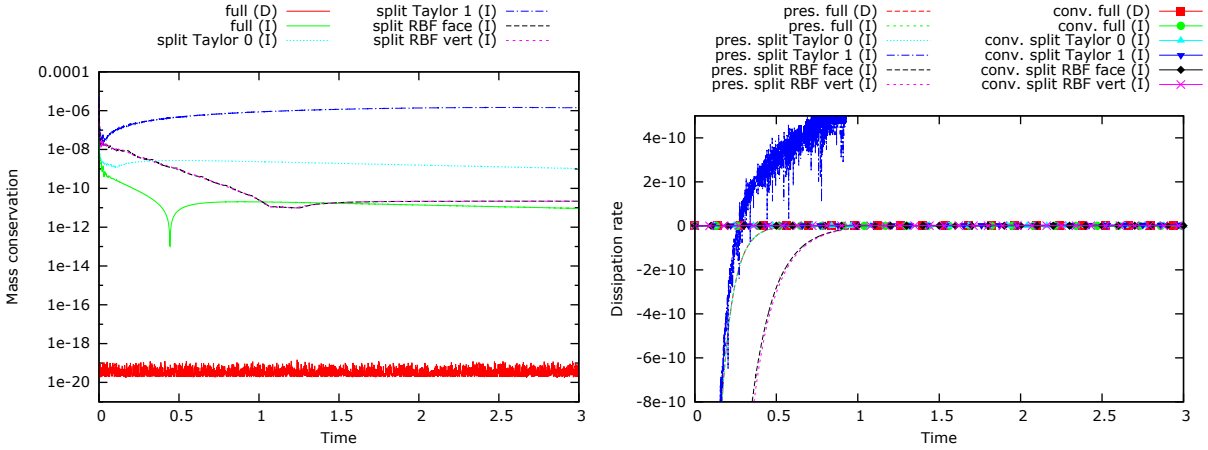


Figure 3: Mass and kinetic energy conservation for different interpolation schemes.

in the mirror node is interpolated using any interpolation scheme for unstructured data $\phi_m^n = f\left(\phi_{nb_1}^n, \phi_{nb_2}^n, ..., \phi_{nb_n}^n\right)$, and once the value ϕ_m^n has been calculated, the values for velocity and pressure in the sliding face can be calculated similarly to the inner faces: $\mathbf{u}_f^n = \frac{1}{2}\left(\mathbf{u}_c^n + \mathbf{u}_m^n\right)$, $\hat{U}_f^n = \frac{1}{2}\left(\mathbf{u}_c^n + \mathbf{u}_m^n\right) \cdot \hat{\mathbf{n}}_f$ and $p_f^{n+1} = \frac{1}{2}\left(p_c^{n+1} + p_m^{n+1}\right)$. However, this strategy cannot be used for the implicit Poisson equation. In a static mesh, if Eq. (6) is applied to all the cells a system of type $[A]\mathbf{p^{n+1}} = \mathbf{b}$ is obtained, where matrix $[A]$ only depends on the geometry and therefore does not change during simulation time. In the present case this matrix $[A]$ is not constant, and is reconstructed at each iteration using Eq. (8) for all the faces in the sliding boundary. Basically, the main idea is that the neighbour cell of cell c is the cell where the mirror node is living in the current time step.

$$\frac{1}{2}\left(\mathbf{u}_c^p + \mathbf{u}_m^p\right) \cdot \hat{\mathbf{n}}_f A_f = \frac{\Delta t}{\rho V_c}\left(p_{nb1}^{n+1} - p_c^{n+1}\right)\frac{A_f}{\delta d_f} \tag{8}$$

2.2 Conservation analysis

The objective of the present section is to study and analyse the conservative properties of the implemented methodology. As demonstrated by Jofre et al. [2], when the symmetry-preserving scheme is employed for the convective scheme, momentum and mass are numerically conserved, except for an error of the form $\mathcal{O}(\Delta t^m, \Delta h^2)$ due to the pressure term in collocated formulation. The problem chosen is a spatially periodic case of vortices described by equations $u = -C\sin(kx)\cos(ky)\,e^{-2k^2\nu t}$ and $v = C\cos(kx)\sin(ky)\,e^{-2k^2\nu t}$, where $C = 3.2 \times 10^{-3} m/s$ is the velocity amplitude, $k = 1$ is the wave number and the kinematic viscosity is set to zero ($\nu = 0$) in order to eliminate the effects of the diffusive term. The problem is solved in a 2D mesh of size $2\pi \times 2\pi$ and the density is set to unity $\rho = 1$. The conservation properties are a function of different simulation parameters, mainly the interpolation scheme employed to calculate ϕ_m, the time step Δt, the rotating velocity ω and the mesh size Δh. Therefore, a parametric study has been carried out for each one in order to characterise its influence in the conservative behaviour of the method.

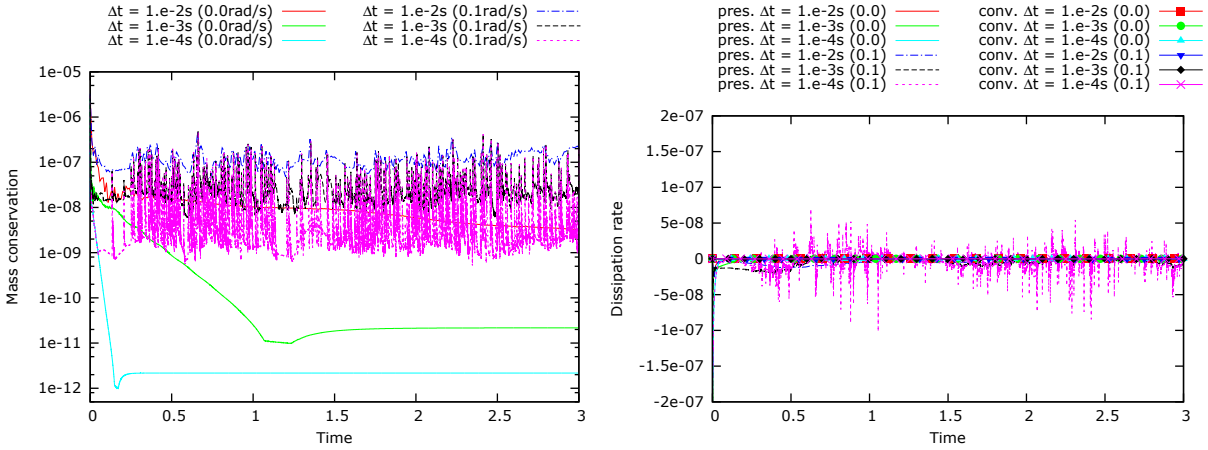


Figure 4: Mass and kinetic energy conservation for different Δt.

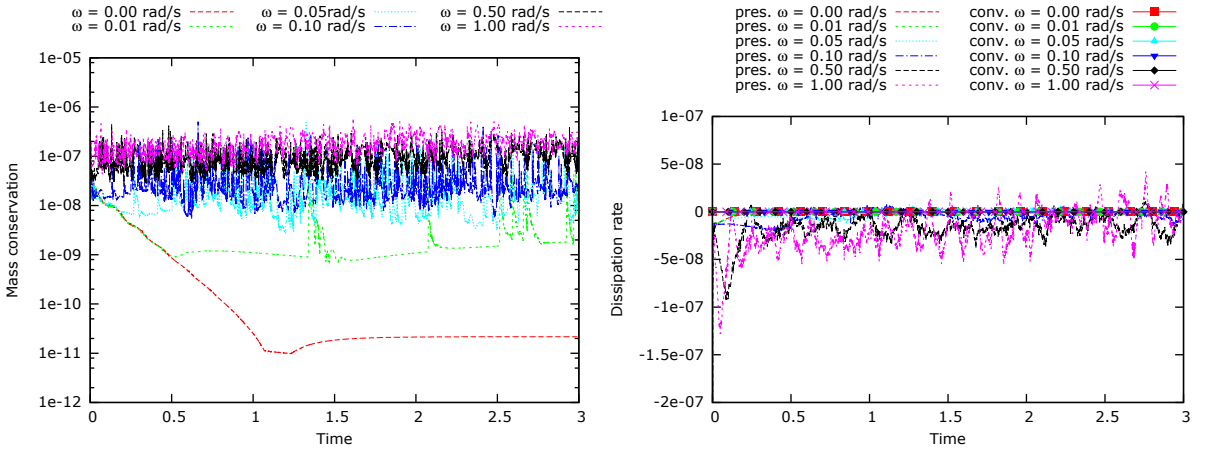


Figure 5: Mass and kinetic energy conservation for different rotating velocities ω.

The conservation analysis for different interpolation schemes is depicted in Fig. 3. The conservation of a mesh without sliding boundary (full mesh) is compared using a direct solver (Direct Schur-complement based decomposition (DSD) [5]) and an iterative solver (Conjugate-Gradient (CG) method with Jacobi diagonal scaling) for solving the Poisson Eq. (6), since the presented method is only able to work with an iterative solver. The interpolation methods compared are the zero and first order Taylor, and the Normalized Radial Basis Function method (Shepard [3]) with a stencil of cells connected by faces and a stencil of cells connected by vertices. The methods are tested in a mesh with sliding boundary and null rotating velocity ($\omega = 0\, rad/s$) and as can be seen, the RBF methods are able to obtain the same order of conservation error than the ones obtained in a mesh without sliding boundary. Therefore, the following conservation tests have been carried out with the RBF with vertex connectivity.

In Fig. 4 the conservation properties are analysed for different time steps and two rotating velocities. As expected, the mass conservation is better for smaller time steps.

Notice that for the rotating mesh ($\omega = 0.1\,rad/s$) the conservation error difference between time steps is smaller than for the static case ($\omega = 0\,rad/s$). The kinetic energy conservation error is in the same order of magnitude for all the cases. A similar trend is found in Fig. 5 where the effect of varying the rotating velocity (keeping constant the time step) is further studied. If the rotating velocity increases, the mass conservation is reduced. However, the mass conservation error seems to collapse at a certain value when increasing the rotating speed.

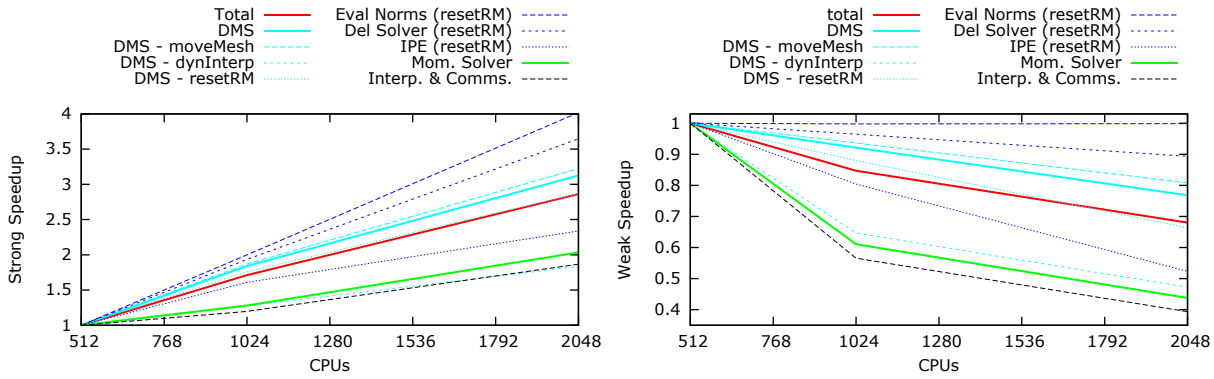
2.3 Parallelilzation strategy and speedup

As described previously, the mirror nodes are lagrangian particles that are going to move during the simulation time. The mirror nodes probably will change of CPU several times along their trajectory, therefore, a robust and efficient strategy of parallelilzation is required. Following, the parallelilzation strategy is explained:

Each injected mirror node stores the following information: its current position, the Cell ID where is living at the current time step, the parent Cell ID, and the parent Rank ID. Since a CPU knows how many mirror nodes has injected (basically, the number of parent cells that the CPU owns) and each mirror node knows his parent rank, the communication strategy at every iteration is as follows: i) Each CPU runs through all its mirror nodes performing the required interpolations, ii) The interpolated data together with the parent Cell ID that is going to receive the information is stored in buffers with size $sizeBuf2Send_i$ (where subindex i refers to the parent Rank ID). iii) An MPI_Alltoall($sizeBuf2Send_i$,...,$sizeBuf2Rcv_i$,...) communication is performed. After this step, each CPU knows how many data have to send and receive from all the CPUs. iv) The buffers are communicated using point to point communications between the processors with $(sizeBuf2Send_i = sizeBuf2Rcv_i)! = 0$. In order to enhance the performance a special communicator is created at the beginning of the simulation time, including only the CPUs that have cells with a face in the sliding boundary and CPUs that are candidates to contain mirror particles at some point of the simulation time.

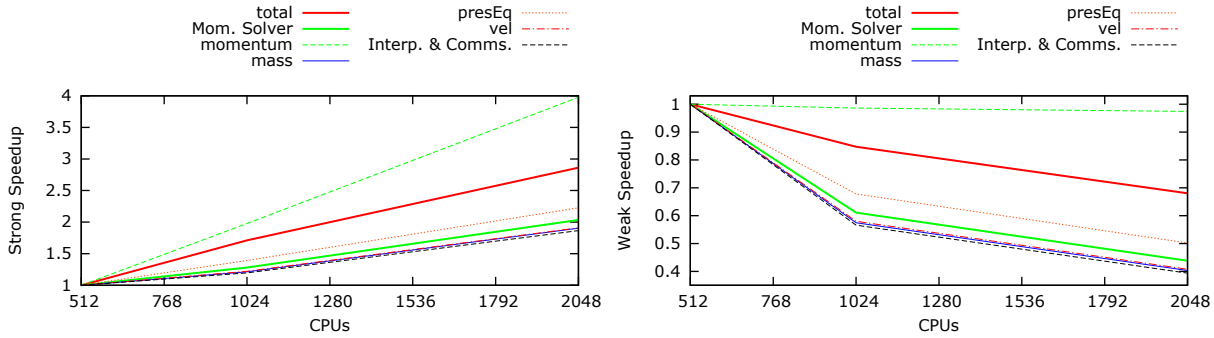
Strong and weak speedup tests have been carried out in order to test the parallel efficiency of the algorithm. These scalability tests have been done solving the canonical Driven Cavity problem. For the strong speedup a mesh of approximately $10.7M$ cells have been employed in 512, 1024 and 2048 CPUs, while the weak speedup have been computed starting in 512 CPU with the same mesh than for the strong speedup case, increasing the number of planes for the 1024 and 2048 CPU cases. These speedup tests have been carried out in the Vesta Cluster of the Argonne Leadership Computing Facility. The Vesta system configuration is based in an IBM BG/Q architecture, and has 2048 nodes with 16 1600 MHz PowerPC A2 cores per node (total cores: 32768). The memory per node is 16 GB RAM and the interconnection is done via a 5D Torus Proprietary Network.

The results for the strong and weak speedup are decomposed into different sub-steps of the algorithm (see Fig. 6). The two main steps are: the Dynamic Mesh Step (*DMS*) and the Momentum solver. The *DMS* includes all the steps related with the displacement of the mesh: the movement of the mesh (*moveMesh*), the displacement of the mirror nodes



(a) Strong speedup. (b) Weak speedup.

Figure 6: Detailed speedup.



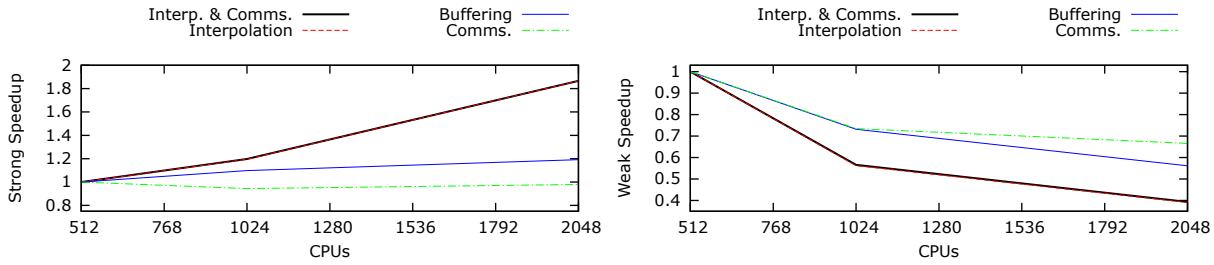
(a) Strong speedup. (b) Weak speedup.

Figure 7: Momentum solver speedup.

(*dynInterp*) and all the tasks required for the new mesh (*resetRM*), like re-evaluating the face normals, delete the solver linked with the old topology, and construct the new topology for the pressure equation (*IPE*). The Momentum step (Fig. 7) is decomposed in the evaluation of the convective and the diffusive operators (*momentum*), the evaluation of mass fluxes at faces (*mass*), the calculation of the velocities at the cell nodes (*vel*) and the resolution of the pressure equation (*pressEq*). Moreover, the steps of interpolation and communication related with the implemented methodology are also shown in Fig. 8.

The total running time for the baseline case (512 CPUs) is 1660 seconds (200 iterations). More specifically, the computational time for the *DMS* was 1375 seconds (82.8%) while the Momentum step was 285 seconds (17.2%). The time spent by the substeps of the *DMS* were: 1062 seconds (63.98%) for the movement of the mesh (*moveMesh*), 6 seconds (0.36%) for the displacement and tracking of the mirror nodes (*dynInterp*), and 307 seconds (18.49%) for all the subtasks involved in the reset of the mesh (*resetRM*).

The substeps with a lower speedup are the reconstruction of the Poisson (*IPE*), the interpolation and the communications (*Interp. & Comms.*), and the translation of the lagrangian particles (*dynInterp*). All these steps have a limited speedup because they are



(a) Strong speedup. (b) Weak speedup.

Figure 8: Interpolation & Communications speedup.

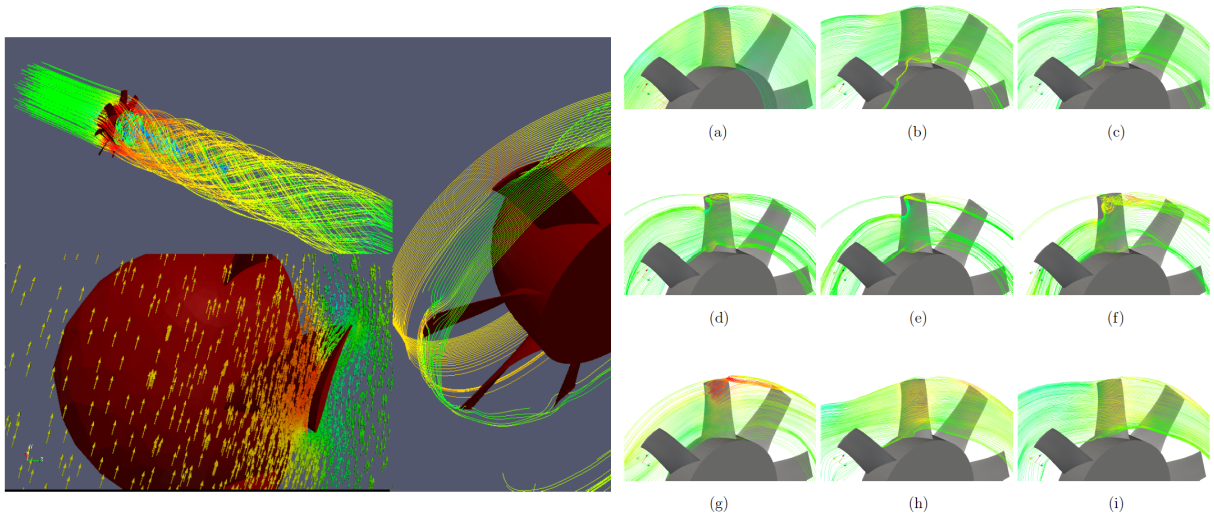


Figure 9: EFFAN Simulation.

strongly linked to the distribution of the mirror nodes in the mesh. Depending on the mesh partitioning and the position of the sliding boundary $\partial\Omega_i$ in the domain $\Omega = \Omega_a \cup \Omega_b$, most of the mirror nodes could live in a few number of CPUs, unbalancing the computational effort and limiting the speedup of the steps involving the mirror nodes, as can be seen in the current example. Nevertheless, the total scalability is quite reasonable and promising, since the strong speedup for the 1024 CPUs case is 1.71 (parallel efficiency: 85.6%) and for the 2048 CPUs case is 2.86 (parallel efficiency: 71.56%).

In the Momentum solver step (Fig. 7), the evaluation of the convective and the diffusive operators scales perfectly, since it is a sequential step that not need any communications. The substeps for mass and velocity evaluation have a speedup that is strongly linked to the Interpolation & Communications speedup, since interpolations in the mirror nodes and communications between the meshes are required in these steps. Regarding the Interpolation & Communications, although the buffering and communications seems not to scale very well, their weight is negligible, since almost all the computational effort is devoted to the interpolation step. As a matter of fact, for the baseline case, the

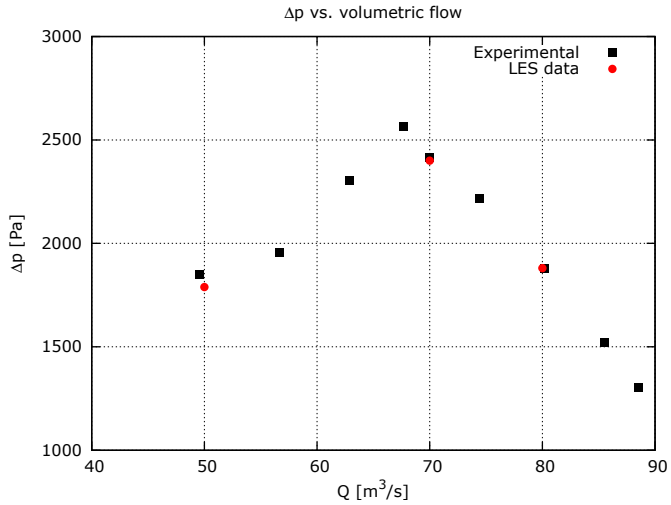


Figure 10: Simulation results compared against experimental data.

Interpolation step took 230 seconds, while the time spent together for the Buffering and the Communications steps was less than a second. As can be seen, the speedup of the Interpolation & Communications is practically the same as the interpolation step speedup, and the interpolation step speedup is strongly dependent on the mesh partitioning and the distribution of the mirror nodes.

3 REAL TEST CASE

In the following section an industrial application of the introduced methodology is presented. Concretely, the aim is the development of an alternative ram-air fan lay-out for the **More Electrical Aircraft (MEA)**. This task is inside the ongoing research project **EFficient FAN (EFFAN)**. The EFFAN project is being developed in the context of CleanSky project, in particular in the Systems for Green Operation (SGO) ITD. The new fan shall be capable to generate pressure drop whatever the flow without surge issues.

The simulations have been carried out employing the second order conservative discretization for unstructured meshes presented in section 2.1 and Large Eddy Simulation (LES) modelling, with the unresolved Reynolds stresses closed by means of a Wall-adapting eddy viscosity model (WALE) [4] SGS model. The cases have been computed in unstructured meshes from 100k to 10M control volumes in 512 CPUs of the UPC JFF cluster (Technical specifications: 40 cluster nodes, each node has 2 AMD Opteron with 16 Cores for each CPU linked with 64 Gigabytes of RAM memory and an infiniband QDR 4X network interconnection between nodes with latencies of 1.07 microseconds with a 40Gbits/s bandwith.). As can be seen in in Fig. 10, the simulation results obtained using the presented method agree very well with the experimental data of the pressure rise across the fan rotor provided by the fan supplier.

4 CONCLUSIONS

In the current work a new methodology for CFD simulations with adjacent disconnected unstructured 3D meshes has been presented, studied and assessed. Moreover, the methodology has been applied to a demanding real test case and has demonstrated to be a powerful tool for simulations of fans, turbo-machinery, wind turbines, etc.

Acknowledgements. This work has been financially supported by the Ministerio de Educación y Ciencia, Spain, (Project: "Desarrollo de códigos y algoritmos paralelos de altas prestaciones para la mejora de la eficiencia en los sectores eólico, solartérmico y edificación", reference ENE2014-60577-R), the CleanSky Project "EFFAN-Efficient Fan" (Grant Agreement no. 620129) and with the support of the Departament d'Innovació, Universitat i Empresa and Comissionat per a Universitats i Recerca de la Generalitat de Catalunya and the European Social Fund.

REFERENCES

[1] Trias, F.X., Lehmkuhl, O., Oliva, A., Pérez-Segarra, C.D., Verstappen, R.W.C.P., Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured grids. *Journal of Computational Physics* (2014), vol. 258, pag. 246-267.

[2] Jofre, L., Lehmkuhl. O., Ventosa, J., Trias, F.X. and Oliva, A., Conservation Properties of Unstructured Finite-Volume Mesh Schemes for the Navier-Stokes Equations. *Numerical Heat Transfer, Part B* (2014), vol. 65-I, pag. 53-79.

[3] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery., Numerical Recipes 3rd Edition: The Art of Scientific Computing (3 ed.). Cambridge University Press, New York, NY, USA. 2007.

[4] Nicoud, F. and Ducros, F., Subgrid-scale stress modeling based on the square of the velocity gradient tensor. *Flow, Turbulence and Combustion* (1999), vol. 62, pag. 183–200.

[5] Borrell, R., Lehmkuhl, O., Trias, F.X. and Oliva, A., Parallel direct Poisson solver for discretisations with one Fourier diagonalisable direction. *Journal of Computational Physics* (2011), vol. 230(12), pag. 4723–4741